AD-A232 644

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final Report, 01 Mar 88 to 30 Apr 90

4. TITLE AND SUBTITLE
SOFTWARE REVERSE ENGINEERING

5. FUNDING NUMBERS
AFOSR-88-0116
61102F 2304/A2

6. AUTHOR(S)
Professor NoahS. Prywes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
School of Engineering and Applied Science
Dept of Computer & Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia, PA 19104-6389

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 91 0 [illegible]

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

DTIC
ELECTE
MAR 08 1991
S E D

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFOSR-88-0116

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of this research was to develop algorithms to translate procedural languages (such as FORTRAN and PASCAL) to non procedural languages. Such algorithms are desirable for a number of reasons. Their semantics is easier to understand since they resemble a set of specification for action rather than the traditional flow of control. They are easier to modify since statements are not dependent on one another. It is easier to verify their correctness since standard theorem provers used for program verifications require programs to be in non procedural form. Lately, with the increase need in DoD of translating old programs to new languages such as Ada (reverse software engineering). It is advantageous to reduce old program to a common non procedural form before translation from that form to the target language. This will permit the recapture of the program's semantics, make desired modificaton, check its correctness, etc. Accomplishments are 1) algorithms to translate FORTRAN programs to MODEL. FORTRAN representing a procedurel language and MODEL represent a non procedural declarative languate. 2) Algorithm to translate concurrent FORTRAN programs to MODEL.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

UNIVERSITY of PENNSYLVANIA

School of Engineering and Applied Science
Department of Computer and Information Science
200 South 33rd Street
Philadelphia, PA 19104-6389

January 11, 1991

Dr. A. Waksman
Air Force Office of Scientific Research
Bolling Air Force Base, DC 20332

Re: Contract #AFOSR–88–0116

Dear Dr. Waksman:

Please accept this letter as the Final Report for Contract AFOSR–88–0116 titled, "Software Reverse Engineering". Research was conducted from March 1988 until April 1990 and the funding was $171,970.

The discussion below describes the goals of the research, the accomplishments and future research and development to exploit the accomplishments.

1. The Goals of the Research

The research focused on developing two–way automatic translation between two classes of languages in which computer programs can be represented:

- procedural languages used widely for programming, and
- nonprocedural mathematical languages also called functional, assertive, etc. being used in a new paradigm for programming.

Mathematical languages have simpler semantics and can be manipulated by a powerful algebra and therefore are claimed to be easier to use, understand and verify.

Manual translations from procedural to mathematical languages are used in a number of methodologies, ranging from program verification to parallel programming. Automating the translation will provide strong impetus to these methodologies. The great importance of such an automatic capability has recently become evident due to the increasing needs for software reverse engineering – the conversion of old programs to new languages (Ada) and new platforms. A two–way translation – of old procedural programs to a mathematical language and hence to the new procedural language and to a new platform – is in effect the embodiment of software reverse engineering. For these reasons, the capability to accomplish the translation automatically will have a major impact on developing improved software development paradigms.

The goal of the project was divided into two steps:

1. Developing a translation algorithm for purely sequential programs (typically used in scientific applications).
2. Developing a translation algorithm for concurrent programs (essential for real–time applications).

The MODEL language was used as an example of nonprocedural mathematical languages. It can be manipulated by regular and boolean algebras. A translation from MODEL to procedural languages (PL/1, C and Ada) has been developed at the University of Pennsylvania over the past decade. Thus the project focused on the reverse translation: from procedural languages to MODEL.

2. Accomplishments

The project accomplished the above goals.

Algorithms were developed for the translation in two steps as follows.

1. **Algorithm for translation of high level pure sequential procedural language (e.g. Fortran) to the MODEL language:** The algorithm is lengthy and complex. Several of the staff examined and hand simulated the algorithm and suggested modifications and extensions. For this reason a number of revisions of the report of this algorithm were issued.

   The translation consists of eight transformations. Starting with the source program, each of a first group of six program transformations modifies a program into an equivalent program which is progressively closer to the mathematical language. The source program is first translated into a basic subset of constructs. It consists of variable declarations (including types and structures), blocks (ifs and loops) and assignments. The basic statements are general purpose and typical of high level programming languages. The declarations, blocks, and assignments have a common meaning in programming languages, although the syntax differs. "Goto's" are translated in "whiles' Procedure calls are viewed as operations on their parameters. Each of the first six transformations provides as output a program that is equivalent to the input to the transformation. It starts with the source program and ends with a single assignment, single value variable program that can be directly transformed into mathematical equations and declarations.

   There is a second group of transformations that make the specification easier to read, understand and modify. The objective is not only that the specification is equivalent to the source program, but also that it be readable and understandable. The transformations in the second group use algebraic symbolic manipulations to operate on the equations. They collect like factors to simplify logical expressions and use substitutions to reduce the number of variables.

2. **Algorithm for translation of concurrent procedural language to the MODEL language:** This algorithm was reported toward the end of the project. It consists of an extension to the previously developed algorithm, to handle concurrent programs with shared memory.

   Concurrent programming examples were used to investigate the reverse translation of typical real–time procedures that share memory with other tasks. We also developed graphic visualizations in the form of dependency diagrams. Evaluation results are displayed in a table to further explain the specification and outline the reasoning about it.

   Shared variables were viewed similar to external inputs and outputs. They provide an interface between concurrent programs. Every reference to a shared variable value is viewed as if it is newly read or written from, or to, an i/o device, and is given a different name. In this way we can consider the shared memory with each of the concurrent programs independently, instead of having to consider all the concurrent programs simultaneously.

Publications

Reports

1. N. Prywes, X. Ge, I. Lee and M. Song, "Reverse Software Engineering", Technical Report, Revision 3, Contract AFOSR–88–0116, submitted to the Air Force Office of Scientific Research, Bolling Air Force, Maryland 20332, December 1989.
   This report documents the algorithm for translating sequential procedural language programs to a mathematical language.
2. M. Song, "Reverse Software Engineering of Concurrent Real Time Programs", Technical Report, Contract AFOSR–88–0116, submitted to Air Force Office of Scientific Research, Bolling Air Force Base, Maryland 20332, December 1989.
   Also Master of Science Thesis in Computer Science, University of Pennsylvania.
   This report documents the extension of the translating algorithm for concurrent programs.

Journal

1. N. Prywes, X. Ge., I. Lee and M. Song, "Procedural to Mathematical Language Translation of Computer Programs".
   Accepted for publication by *Science of Programming*, to be published in 1991.

Conferences

1. X. Ge, N. Prywes "Reverse Software Engineering", Proceeding of the Workshop on Reverse Engineering, Naval Surface Warfare Center, White Oak, Maryland, 20903–5000, April 1990.
2. N. Prywes, "Reverse Software Engineering of Concurrent Programs", Proceeding of Knowledge Base Software Assistant Conference, Rome Air Development Center, Rome, NY, August 1990.

Invited Talks

The results of the project were presented in a number of invited presentations at Conferences (National CASE Conference, 1990) Universities (University of Pennsylvania, Stanford University) and Laboratories (Avionics, Wright–Patterson Air Force Base; Navy Ocean Systems Command).

3. Suggested Future Research

As noted, the developed algorithms can be used for software development in the re–engineering paradigms. The use of the translation algorithms for software reverse engineering was discussed above.

The representation of a program in a mathematical language allows manipulation using regular and boolean algebraic semantics. It can be used in far reaching ways in a number of methodologies ranging from program verification to parallel programming.

There is a potential for significant advances in software understanding through use of the mathematical languages instead of the conventional programming languages. Our reports have discussed the use of graphics for presenting the algorithms of the software. We proposed a software understanding system that will utilize these graphs for software visualization. These graphs will be displayed to the user. They will be used for understanding, modification, verification and testing. This approach will make use of Artificial Intelligence Expert Systems or/and Symbolic Manipulation Systems to explain the software to the user.

Sincerely yours,

Noah Prywes

Noah S. Prywes
Principal Investigator
Professor of Computer Science

NSP:tc